Robust Parametric Reachability for Timed Automata

Laurent Doyen [1]

School of Computer and Communication Sciences
Swiss Federal Institute of Technology (EPFL)

Abstract

We review the known decidability and undecidability results for reachability in parametric timed automata. Then, we present a new proof of undecidability in dense time for open timed automata that avoids equalities in clock constraints. Our result shows that the undecidability of parametric timed automata does not follow from their ability to specify punctual constraints in a dense time domain.

Key words: Parametric timed automata, real-time systems, formal methods, model checking, theory of computation, robustness, verification.

1 Introduction

The design and verification of real-time systems has intensively used formal methods for several years. In that context, *timed automata* have played an important role [1]. A timed automaton is essentially a finite automaton augmented with a set of *clocks* that allow to specify timing constraints on the transitions of the automaton. Two models of time are usually considered, either the time domain $\mathbb{T}$ is discrete ($\mathbb{T} = \mathbb{N}$) or dense ($\mathbb{T} = \mathbb{R}^{\geq 0}$ or $\mathbb{T} = \mathbb{Q}^{\geq 0}$).

An important result about timed automata is that the *reachability problem* ('Is a given location of the automaton reachable from the initial state ?') is decidable when the constants that appear in the timing constraints are rational numbers [1]. The problem becomes undecidable when irrational constants are allowed [10].

In this paper, we are interested in the reachability problem for *parametric* extensions of timed automata, where constants in timing constraints are replaced by parameters representing unknown constants. The possibility to specify parametric constraints allows to design systems independently of a particular implementation. For example, the speed of the hardware or the time transmission in a communication protocol could be left as parameters in the early phases of the design. The *parametric reachability problem* asks then whether there exists a valuation for the parameters such that a given location is reachable. We review the known (un)decidability results about this problem when the domain $\mathbb{P}$ of the parameters is either discrete ($\mathbb{P} = \mathbb{N}$) or dense ($\mathbb{P} = \mathbb{R}^{\geq 0}$ or $\mathbb{P} = \mathbb{Q}^{\geq 0}$). The main result is that the problem is undecidable when $\mathbb{T}$ and $\mathbb{P}$ are dense [4, 7].

The classical proofs of that result use the expressiveness of equality in timed automata to encode Turing-machine computations. Similar results exist for real-time logics. However, if the use of equality is disallowed in the logic, decidability can often be established. This is the case for MITL [3] and the parametric extension of TCTL [6]. The hope has then arisen that decidability of the parametric reachability problem could be established for more robust models like *open timed automata* that avoid equality constraints. Unfortunately, we show that this is not the case and that parametric timed automata are *robustly* undecidable. The proof that we present is the main contribution of the paper. It is based on a non-trivial novel encoding of two-counter Minsky machines.

2 Parametric Timed Automata

Given a set Var of clocks and a set P of parameters, let $\Phi(\mathsf{Var}, \mathsf{P})$ be the set of *clock constraints* φ defined by the grammar rule $\varphi ::= x \sim c \mid \varphi \wedge \varphi$ where $x \in \mathsf{Var}$, $c \in \mathbb{N} \cup \mathsf{P}$ and $\sim \in \{<, \leq, =, \geq, >\}$.

Definition 1 [PTA - Parametric Timed Automata] A *parametric timed automaton* is a tuple $A = \langle \mathsf{Loc}, \ell_0, \mathsf{Var}, \mathsf{P}, \mathsf{Lab}, \mathsf{Edg}, \mathsf{Inv} \rangle$ where : (i) Loc is a finite set of *locations*; (ii) $\ell_0 \in \mathsf{Loc}$ is the initial location; (iii) Var is a finite set of variables called *clocks*; (iv) P is a finite set of *parameters*; (v) Lab is a finite alphabet of *labels*; (vi) $\mathsf{Edg} \subseteq \mathsf{Loc} \times \mathsf{Loc} \times \Phi(\mathsf{Var}, \mathsf{P}) \times \mathsf{Lab} \times 2^{\mathsf{Var}}$ is a set of *edges*. An edge $(\ell, \ell', g, \alpha, R) \in \mathsf{Edg}$, represents a transition from location ℓ to location ℓ' with *guard* g, label α and a subset $R \subseteq \mathsf{Var}$ of the clocks to be reset; (vii) $\mathsf{Inv} : \mathsf{Loc} \to \Phi(\mathsf{Var}, \mathsf{P})$ is the *invariant* condition. The automaton can stay in location ℓ as long as the current values of the clocks satisfy the constraint $\mathsf{Inv}(\ell)$.

The semantics of PTA is given by a transition system. The states are pairs (ℓ, v) where $\ell \in \mathsf{Loc}$ and $v : \mathsf{Var} \to \mathbb{T}$ is a *clock valuation*. The transition relation depends on the valuation of the parameters. Let $\kappa : \mathsf{P} \to \mathbb{P}$ be a *parameter valuation* and define $\kappa(c) = c$ for every $c \in \mathbb{N}$. For a formula $\varphi \in \Phi(\mathsf{Var}, \mathsf{P})$ and a clock valuation v, we write $v \models_\kappa \varphi$ iff $v(x) \sim \kappa(c)$ for each '$x \sim c$' appearing in φ. Define $[\![\varphi]\!]_\kappa = \{v \mid v \models_\kappa \varphi\}$. Given a valuation v and $t \in \mathbb{T}$, the valuation $v + t$ assigns the value $v(x) + t$ to each variable $x \in \mathsf{Var}$.

$\mathbb{T}$	Clocks compared to parameters	Other clocks	Parameters ($\mathbb{P} = \mathbb{T}$)	Decidability
$\mathbb{N}$	1	any	any	$\surd$ [4]
$\mathbb{R}$	1	0	any	$\surd$ [4, 10]
$\mathbb{N}$ or $\mathbb{R}$	3	0	6	$\times$ [4]
$\mathbb{R}$	3	0	1	$\times$ [10]
$\mathbb{R}$	1	3	1	$\times$ [10]

Table 1. Existing decidability ($\surd$) and undecidability ($\times$) results for PTA.

Definition 2 [Semantics of PTA] Given a parameter valuation κ, the semantics of a PTA $A = \langle \mathsf{Loc}, \ell_0, \mathsf{Var}, \mathsf{P}, \mathsf{Lab}, \mathsf{Edg}, \mathsf{Inv} \rangle$ is given by the *labelled transition system* $[\![A]\!]_\kappa = (S, S_0, \mathcal{L}, \mapsto)$ where : (i) $S = \{(\ell, v) | \ell \in \mathsf{Loc} \wedge v \in [\![\mathsf{Inv}(\ell)]\!]_\kappa\}$; (ii) $S_0 = \{(l_0, v_0)\}$ where $v_0(x) = 0$ for every $x \in \mathsf{Var}$; (iii) $\mathcal{L} = \mathsf{Lab} \cup \mathbb{T}$; (iv) The relation $\mapsto \subseteq S \times \mathcal{L} \times S$ is defined as follows: (a) Discrete transitions. For $\sigma \in \mathsf{Lab}$, $((\ell, v), \sigma, (\ell', v')) \in \mapsto$ iff there exists an edge $(\ell, \ell', g, \sigma, R) \in \mathsf{Edg}$ such that $v \models_\kappa g$, $v'(x) = 0$ if $x \in R$ and $v'(x) = v(x)$ if $x \notin R$. (b) Timed transitions. For $t \in \mathbb{T}$, $((\ell, v), t, (\ell', v')) \in \mapsto$ iff $\ell' = \ell$, $v' = v + t$ and for every $t' \in [0, t] : v + t' \in [\![\mathsf{Inv}(\ell)]\!]_\kappa$.

A state s_f is *reachable* in a labelled transition system $\mathcal{T} = (S, S_0, \mathcal{L}, \mapsto)$ iff there exists a finite sequence $\bar{s} = s_0, s_1, \ldots, s_n$ of states $s_i \in S$ such that $s_0 \in S_0$, $s_n = s_f$ and for every $0 \leq i < n$, there exists some $\sigma_i \in \mathcal{L}$ such that $(s_i, \sigma_i, s_{i+1}) \in \mapsto$. We write $\mathsf{Reach}(\mathcal{T})$ for the set of reachable states of $\mathcal{T}$.

3 Parametric Reachability

Given a PTA A and a location ℓ_f, the set $\Gamma_{\ell_f}(A) = \{\kappa \mid (\ell_f, v_f) \in \mathsf{Reach}([\![A]\!]_\kappa)$ for some valuation $v_f\}$ contains the parameter valuations such that ℓ_f is reachable in A.

Definition 3 Given a PTA A and a location ℓ_f, the *parametric reachability problem* asks whether $\Gamma_{\ell_f}(A)$ is empty.

In Table 1, we give a summary of the existing results about decidability of this problem, depending on the time domain $\mathbb{T}$ and the number of clocks and parameters. We assume that the parameters take their value in the set $\mathbb{T}$, that is $\mathbb{P} = \mathbb{T}$. It would make sense to consider the case $\mathbb{T} = \mathbb{R}^{\geq 0}$ and $\mathbb{P} = \mathbb{N}$, but the problem is obviously undecidable in general since it is already the case for $\mathbb{P} = \mathbb{T} = \mathbb{N}$ (3rd line in Table 1). Notice that all the results presented for $\mathbb{P} = \mathbb{T} = \mathbb{R}^{\geq 0}$ hold for $\mathbb{P} = \mathbb{T} = \mathbb{Q}^{\geq 0}$.

The parametric reachability problem in discrete time is *decidable* for the class of PTA with an arbitrary number of clocks in which only one clock is compared to the parameters [4]. The proof is in two steps. First the non parametrically constrained clocks are eliminated, and then a linear formula defining $\Gamma_{\ell_f}(A)$ is constructed. The decidability of testing emptiness of $\Gamma_{\ell_f}(A)$ follows.

In dense time, decidability is established only for PTA with one single clock and the problem is NP-complete in this case [4, 10]. However, for PTA with four clocks, the parametric reachability problem is undecidable even if only one clock is parametrically constrained [10]. Finally, as stated by Theorem 1, the parametric reachability problem is undecidable in both discrete and dense time for PTA with at least three clocks and six parameters [4].

Theorem 1 ([4]) *The parametric reachability problem is undecidable in dense time for general* PTA.

The proof uses a reduction from the halting problem for 2-counter machine which is known to be undecidable [11]. A *2-counter machine* consists of a finite set of states $Q = \{q_0, \ldots, q_m\}$ (with an initial state q_0 and a final state q_m), a finite set of instructions and two counters C_1 and C_2. An instruction is associated to each machine state, and it can be either (increment) $C_k = C_k + 1$ **goto** q_i, or (decrement) $C_k = C_k - 1$ **goto** q_i, or (zero-testing) **if** $C_k = 0$ **then goto** q_i **else goto** q_j, where $k \in \{1, 2\}$ and $q_i, q_j \in Q$ are machine states. The decrement is not allowed if the counter value is 0. We may assume that a zero-testing is done before every decrement.

A *configuration* of the machine is a triple (q_i, c_1, c_2) where $q_i \in Q$ is a state and $c_1, c_2 \in \mathbb{N}$ are the values of C_1 and C_2 respectively. An execution of the machine is an infinite sequence $\pi = \pi_0\pi_1 \ldots$ of configurations such that $\pi_0 = (q_0, 0, 0)$ and for all $i \geq 0$, if $\pi_i = (q, c_1, c_2)$ then π_{i+1} is obtained as expected, according to the instruction associated to the state q. The *halting problem* for 2-counter machines is to decide if a given machine M has an execution that reaches a configuration (q_m, c_1, c_2) for some values c_1 and c_2.

The reduction presented in [4] uses three clocks and six parameters to encode the value of the two counters, and instructions of the 2-counter machine are translated into operations on clocks. In [10], an original proof is presented for dense time. It works for PTA with three clocks and one parameter. The idea is to use a new undecidability result for irrational timed automata where constants can be irrational instead of integers. Once again, this result is proven by reduction of the halting problem for 2-counter machines. Then, it is shown that the reduction is still applicable if the irrational constants are replaced by a parameter (since a rational parameter can be chosen arbitrarily close to an irrational). Refer to [10] for details. Finally, for the subclass of PTA called L/U automata, the parametric reachability problem is decidable [9].

In the field of parametric real-time verification, there are several works where the parameters are introduced in real-time logics like TCTL [12, 8, 2, 5] or simultaneously in the model and in the logic [6].

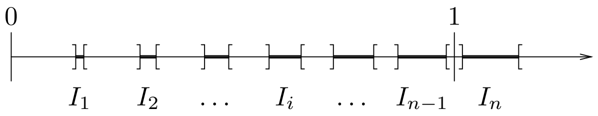

Fig. 1. Intervals of the form $I_i =]i \cdot \alpha, i \cdot \beta[$ in which the clocks x and y lie.

4 Robust Undecidability

In both undecidability proofs of the previous section (Theorem 1), the fact that PTA allow to define strong constraints of the form '$x = \alpha$' where α is a parameter is essential for simulating 2-counter machine. It could be argued that perhaps the parametric decision problem is undecidable simply because equality is too expressive. However, we show that a 2-counter machine can be simulated without using equality. Our reduction technique is inspired by the widget construction presented in [7].

A PTA is *open* if all the guards and invariants are generated by the grammar rule $\varphi ::= x < c \mid x > c \mid \varphi \wedge \varphi$ where $x \in \mathsf{Var}$ and $c \in \mathbb{N} \cup \mathsf{P}$.

Theorem 2 *The parametric reachability problem is undecidable in dense time for open* PTA *(for $\mathbb{P} \in \{\mathbb{R}^{\geq 0}, \mathbb{Q}^{\geq 0}\}$ and $\mathbb{T} \in \{\mathbb{R}^{\geq 0}, \mathbb{Q}^{\geq 0}\}$).*

Proof. Given a 2-counter machine M, we construct an open PTA A_M with five clocks and two parameters α and β. The states $q_0, \dots, q_m$ of the 2-counter machine are encoded by the locations $\ell_1, \dots, \ell_m$ of A_M respectively. The construction is such that the location ℓ_m is reachable for some valuation of the parameters if and only if M halts (or equivalently reaches the state q_m). The value of each counter C_k is encoded by two clocks x_k and y_k of the timed automaton, and we use an additional clock t to generate pseudo-periodical *ticks*: we put the guard $t > \alpha$ on every edge, and we put the invariant $t < \beta$ on every location (except for the automaton of Fig. 2 that we use in an initialization step). Also, we reset t on every edge so that a new tick occurs every between α and β time units with $\alpha < \beta$ (typically, the value of the parameters α and β is intended to be much less than 1).

After i such ticks, a clock x (initially 0) has a value in the interval $I_i =]i \cdot \alpha, i \cdot \beta[$. Now, assume that for some $n \in \mathbb{N}$ (see also Fig. 1):

($A1$) the intervals I_i and I_{i+1} are disjoint for each $0 \leq i < n$;

($A2$) $I_{n-1} \subset [0, 1]$ and $I_n \subset [1, +\infty[$.

Then, if we reset the clock x when $x \in I_n$, that is n ticks after the last reset, we can simulate a modulo-n counter. Now, we use the difference between two such counters to maintain the value of the machine counters as time elapses. Given a maximal constant n, we define the value c of a counter encoded with clocks x and y as follows:

$$\text{if } x \in I_i \text{ and } y \in I_j \text{ then } c = val(i,j) = \begin{cases} i - j & \text{if } i \geq j \\ n + i - j & \text{if } i < j \end{cases}$$

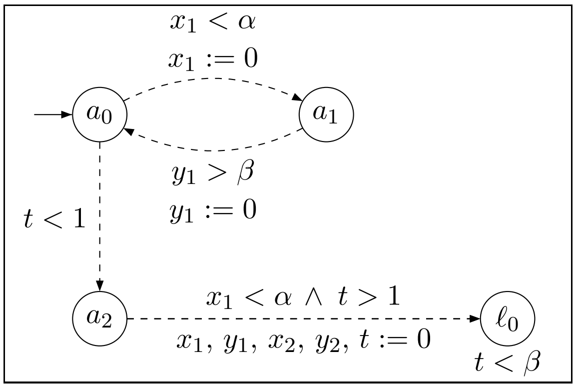

Fig. 2. A^{init}

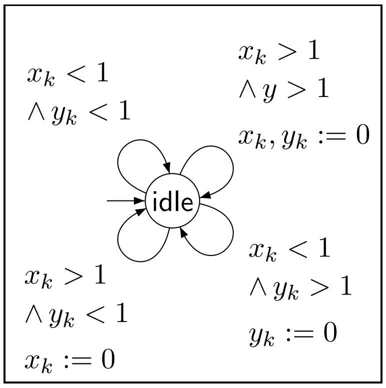

Fig. 3. Idling with A_k^{idle}.

The assumption $(A1)$ guarantees the uniqueness of i such that $x \in I_i$ (and similarly for j). It is easy to establish the following invariance property for this encoding:

$$\forall 0 \leq i, j < n : \ val(i+1 \bmod n, j+1 \bmod n) = val(i,j) \tag{1}$$

Note that the assumption $(A1)$ is equivalent to ask that I_{n-1} and I_n are disjoint (since $\alpha < \beta$ and $(n-1)\beta < n \cdot \alpha$ entails $(i-1)\beta < i \cdot \alpha$ for all $i \leq n$), which is implied by $(A2)$. On the other hand, $(A2)$ is equivalent to the following condition on the parameters: $(n-1)\beta < 1 < n \cdot \alpha$.

We check this condition with the initialization widget A^{init} of Fig. 2 whose location ℓ_0 (corresponding to the machine state q_0) is reachable if and only if there exists $n \in \mathbb{N}$ such that $(n-1)\beta < 1 < n \cdot \alpha$ (in fact n is the number of times the location a_0 is visited before reaching ℓ_0, thus the loop a_0, a_1 is taken $n-1$ times). Observe that the clock x_1 is always reset when $x_1 < \alpha$, so that when the edge (a_2, ℓ_0) is taken, we have $t < n \cdot \alpha$ and $t > 1$ which implies $1 < n \cdot \alpha$. On the other hand, the condition $(n-1)\beta < 1$ is trivially satisfied for $n = 1$. For $n > 1$, since y_1 is reset when $y_1 > \beta$, we have $t > (n-1)\beta$ in the location a_0 when the edge to a_2 is taken with $t < 1$. This entails $(n-1)\beta < 1$.

In this setting, the maximal value of a counter is $n - 1 = \lfloor \frac{1}{\alpha} \rfloor = \lfloor \frac{1}{\beta} \rfloor$. Thus, with a lower value for α we can encode larger values of the counters. Since parameters are valued in $\mathbb{R}^{\geq 0}$, this is sufficient to guarantee faithful simulation of the 2-counter machine, if its counters remain bounded. If a counter overflow occurs in the simulation of M, an error location is reached in A_M where it is impossible to reach ℓ_m. In summary,

- if M reaches q_m, then the values of its counters remain bounded, and so by choosing sufficiently small values for the parameters, A_M will be able to simulate M and thus to reach ℓ_m.

- On the other hand, if M does not reach q_m, then either the counters are unbounded and A_M falls in overflow no matter the choice of the value of the parameters, or A_M can mimic the execution of M forever (as before by choosing sufficiently small values for the parameters), yet cannot reach ℓ_m since M does not.

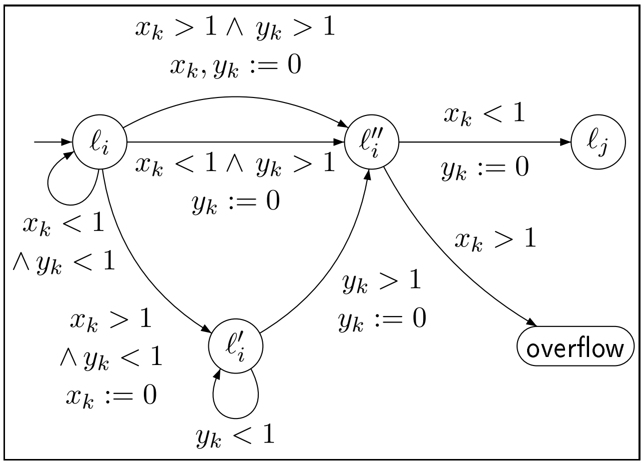

Fig. 4. Incrementing C_k with A_k^+.

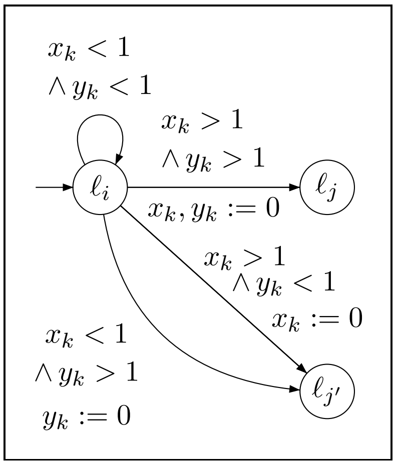

Fig. 5. Zero-testing C_k with A_k^0.

We present the widgets that we use to construct the timed automaton A_M. In all the subsequent figures, the invariant $t < \beta$ on each location, the guard $t > \alpha$ and the reset $t := 0$ on every edge are not depicted for the sake of clarity.

First, consider the idling automaton A_k^{idle} of Fig. 3 (for $k \in \{1, 2\}$). This automaton maintains the value of the counter C_k by resetting x_k and y_k whenever they exceed 1. This widget is used to preserve the value of a counter while executing an instruction involving the other counter. The correctness of A_k^{idle} relies on Equation (1). Now, we show how to execute the three types of instruction of M with A_M.

Increment An instruction of the form $C_k = C_k + 1$ **goto** q_j at state q_i is translated into the synchronized product of $A_k^+$and A_{3-k}^{idle} where A_k^+ is depicted in Fig. 4. It is assumed that all their edges have the same label (not depicted) which ensures synchronizations of the two automata.

We informally explain the structure of A_k^+. Remember that each edge is a tick. The first step is to obtain an encoding of C_k such that $y_k = 0$ in ℓ''_i. To do this, the automaton A_k^+ is idling in location ℓ_i until either $y_k > 1$ or $x_k > 1$. In the first case, we can directly reset y_k and jump to ℓ''_i, and in the second case we have to wait in location ℓ'_i until $y_k > 1$ to do so. The second step is the increment itself. From $\ell''i$, we reset y_k after the next tick and we proceed to ℓ_j. However, if we had $x_1 > 1$ at that time, it would mean that the counter overflows and that the simulation cannot continue. The deadlock location overflow is then reached where no transition is possible.

Decrement A decrement of the form $C_k = C_k - 1$ **goto** q_j at state q_i is translated into the synchronized product of A_k^- and A_{3-k}^{idle} where A_k^- is depicted in Fig. 6. Again, all their edges have the same label.

Decrementing the counter C_k is the dual of incrementing: we proceed to location ℓ''_i when x_k is reset so that $y_k \in I_{n-i}$ if the value of C_k is i. Then, with the next tick, we reset x_k so that $x_k \in I_0$ and $y_k \in I_{n-i+1}$. Thus, the value of C_k is now $i-1$ in ℓ_j. Note that an edge guarded by $x_k > 1 \wedge y_k > 1$ is missing from location ℓ_i in Fig. 6

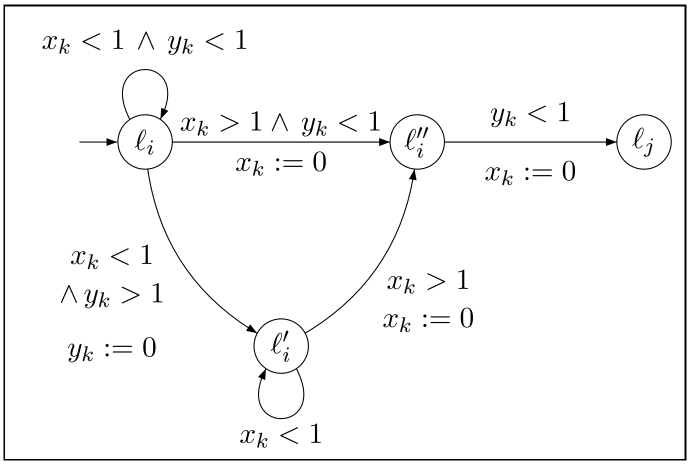

Fig. 6. Decrementing C_k with A_k^-.

since it corresponds to a counter equal to 0 which is prevented by the assumption that counters are zero-tested before decrementing.

Zero-testing An instruction of the form **if** $C_k = 0$ **then goto** q_j **else goto** $q_{j'}$ at location q_i is translated into the synchronized product of A_k^0 and A_{3-k}^{idle} where A_k^0 is depicted in Fig. 5. The value of a counter C_k is zero iff $x_k, y_k \in I_i$ for some i, which means that the two clocks will eventually exceed 1 during the same tick. This is checked by A_k^0, branching to either ℓ_j or $\ell_{j'}$. Again, all their edges have the same label.

The automaton A_M is now built up by concatenating A^{init} and each of the widget translated from the instructions of M. By concatenation, we mean taking the union of the locations and edges of the widgets, with initial location a_0 of A^{init} and final location ℓ_m. It is now clear from the above construction that the following claims are equivalent:

(1) The state (ℓ_m, v) is reachable in $[\![A_M]\!]_\kappa$ for some valuation v.

(2) There exists an execution π' of M containing a configuration (q_m, c_1, c_2) for some $c_1, c_2 \in \mathbb{N}$ and such that for all $i \geq 0$, if $\pi'_i = (q, c_1, c_2)$ then $c_1, c_2 \leq \lfloor \frac{1}{\kappa(\alpha)} \rfloor$.

This allows to conclude that $\Gamma_{\ell_m}(A_M)$ is not empty if and only if the answer to the reachability problem for M is YES, and thus the parametric reachability problem is undecidable for open PTA. $\square$

The proof that we have presented uses five clocks and two parameters, but three clocks are compared with parameters, namely x_1, y_1 and t. It is clear that the same reduction holds with only two clocks compared with parameters: in the initialization widget A^{init}, since all the clocks are reset before entering ℓ_0, we could swap for example x_1 and t so that x_1 is no more compared to parameters.

5 Conclusion

We have seen that introducing parameters in the model of timed automata yields a parametric version of the reachability problem that is undecidable in dense time except for some very restricted cases, with few interactions with parameters (and only one clock in dense time). We have strengthened this result with a new proof of undecidability for parametric timed automata. Unlike the classical proofs of undecidability, our proof does not rely on the use of equality in timing constraints and thus it is more robust. Formally, it applies to *open* parametric timed automata, with at least two parameters and five clocks (among which two are compared with the parameters). It is an open question whether this number of clocks and parameters is tight for undecidability of the reachability problem.

Acknowledgments The author thanks Jean-François Raskin for suggesting the idea of the main proof.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for "model measuring". In *Proc. of ICALP 99: Automata, Languages and Programming*, LNCS 1644, pp. 159–168. Springer-Verlag, 1999.

[3] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. In *Journal of the ACM*, vol. 43, pp. 116–146, 1996.

[4] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.

[5] Véronique Bruyère, Emmanuel Dall'Olio, and Jean-François Raskin. Durations, parametric model-checking in timed automata with presburger arithmetic. In *Proceedings of STACS 03*, LNCS 2607. Springer-Verlag, 2003.

[6] Véronique Bruyère and Jean-François Raskin. Real-time model-checking: Parameters everywhere. In *Proceedings of FSTTCS 03*, LNCS 2914, pp. 100–111. Springer-Verlag, 2003.

[7] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *HSCC'02*, LNCS 2289, pages 134–148. Springer-Verlag, 2002.

[8] E. Allen Emerson and Richard J. Trefler. Parametric quantitative temporal reasoning. In *Proc. of LICS*, pp. 336–343. IEEE Computer Society Press, 1999.

[9] T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager. Linear parametric model checking of timed automata. In *J. Log. Algebr. Program.*, vol. 52-53, pp. 183–220, 2002.

[10] J.S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *HSCC '00*, LNCS 1790, pages 296–309, 2000.

[11] N.M. Minsky. *Finite and Infinite Machines*. Prentice-Hall, 1967.

[12] Farn Wang and Pao-Ann Hsiung. Parametric analysis of computer systems. In *Proc. of AMAST 97*, LNCS 1349, pages 539–553. Springer-Verlag, 1997.